LA RUBRIQUE *POINT DE VUE*

Un lieu de débat pour les enseignants de Mathématiques

La rubrique « POINT DE VUE » est destinée à être un lieu de débat et un outil de réflexion pour les enseignants de mathématiques sur tous les sujets qui concernent leur profession.

Elle accueille dans ce numéro une réflexion de Valéry Bruniaux, professeur en lycée de Physique-chimie et d'ICN/ISN, ingénieur en informatique (ENIB) et membre du groupe Irem « Enseignement de l'Informatique » d'Aix-Marseille

Valéry Bruniaux s'interroge sur le choix d'un langage de programmation pour l'apprentissage de l'Informatique au lycée.

Cette rubrique est ouverte à tous et destinée à recevoir des articles courts, d'environ trois pages…

Nous attendons vos propositions.

Le Comité de Rédaction

Point de vue

CHOIX RAISONNE D'UN LANGAGE DE PROGRAMMATION POUR L'APPRENTISSAGE DE L'INFORMATIQUE AU LYCEE

Valéry BRUNIAUX
Irem d'Aix-Marseille

État des lieux

Une évidence herpétologique

A l'heure actuelle, force est de constater que les choix des enseignants qui souhaitent faire programmer leurs étudiants est restreint à une seule possibilité : Python. Les programmes et les professeurs de Maths de Lycée le recommandent à 101%, nominativement ou en faisant une description du langage idéal d'apprentissage calqué sur les caractéristiques du célèbre serpent, ce qui ne laisse planer aucun doute sur le choix à faire...

Le manque de débat et de réflexion

On pourrait donc se dire qu'un tel plébiscite est certainement fondé en raison et qu'il n'y a pas lieu de remettre en cause ce diktat... Sauf que...

Si on cherche à creuser un peu le sujet on s'aperçoit que ce "choix" n'en est pas un, puisqu'on n'en trouve nulle part une explication rationnelle ni même un débat nourri autour de ce sujet important.

Un choix crucial

En effet, c'est un débat crucial, car même si avec une certaine expertise on comprend que tous les langages procèdent, au final, globalement d'une même logique, il n'en reste pas moins que le langage d'apprentissage est le premier contact de l'élève avec la matière. Il y a donc un enjeu évident d'impression initiale (dans les deux sens du terme).

En effet, commencer l'informatique par du langage machine ou du C++ ne donnera pas au débutant, ni les mêmes approches, ni les mêmes connaissances, ni les mêmes motivations.

Ouroboros fait l'autruche

La meilleure explication à cet état de fait n'est pas politiquement correcte à énoncer mais elle se vérifie presque systématiquement sur le terrain. La plupart des enseignants à qui l'institution a demandé de faire de la programmation avec les élèves ne sont pas des informaticiens. Il se sentent donc véritablement en danger sur ce sujet qu'ils maîtrisent mal, sachant qu'ils peuvent se retrouver face à des élèves plus à l'aise qu'eux-mêmes.

Du coup, les enseignants s'arcboutent sur le seul langage qu'ils connaissent un peu grâce souvent à une légère formation et pour lequel ils ont des ressources, refusant catégoriquement, pour la plupart, d'envisager d'autres langages possibles. Ça ne fait pas honneur à notre corps enseignant mais c'est tout à fait réel, humain et naturel.

Une réflexion possible

Plutôt que de suivre une habitude grégaire et non réfléchie, à l'heure où l'informatique va être proposée massivement au lycée, nous avons à construire une réflexion sur le bon choix du langage de programmation pour prendre le relais des langages "block" utilisés au collège.

Avant de commencer une ébauche de réflexion pour lancer ce débat, précisons que nous cherchons un langage qui aide l'élève à se former à la pensée Informatique et pas nécessairement à une technologie utile à moyen terme, même si cela pourrait être un plus.

Quel paradigme de programmation ?

Avant de choisir un langage, il convient de prime abord d'analyser quel "type" de langage sera adapté à cet apprentissage. En effet, il existe différentes manières de penser un problème informatique. Citons les paradigmes les plus connus : programmation objet (C++, Java, Smalltalk), programmation fonctionnelle (Haskell, Ocaml), programmation impérative (C).

Allons vite sur ce sujet car il faut une bonne expertise de ces domaines de programmation pour en saisir les impacts. Pour résumer, la programmation impérative est la plus utilisée et probablement la plus intuitive puisqu'elle consiste à donner à la machine les instructions à exécuter (tests, affectations, etc.).

La programmation objet, très à la mode dans les années 90, peut tout à fait être enseignée à posteriori et n'est pas forcément indiquée pour débuter. En effet, quand on commence l'apprentissage, les programmes sont courts et la sémantique objet n'est pas spécialement bien adaptée dans ce cas. De plus, le modèle "objet" laisse maintenant sa place à des langages dans lesquels les flux de données sont traités comme des filtrages.

En revanche, la programmation fonctionnelle pourrait être une excellente école pour former les esprits à la pensée informatique. D'abord, parce qu'elle correspond bien à la notion de fonction mathématique, et surtout car elle empêche tous les effets de bord en bannissant la notion d'affectation. Ainsi les programmes sont plus sûrs et plus faciles à vérifier et à maintenir.

Ceci dit, de nombreux langages sont "multi-paradigmes" et ils permettent de penser des programmes de différentes manières. Le paradigme fonctionnel me séduit tout-à-fait pour débuter, mais il me semble faire appel à un niveau de conceptualisation plus poussé. Il me paraît donc plus raisonnable de choisir, pour motiver le plus grand nombre, un langage impératif ou "multi-paradigmes". Le Python reste donc en course ;-).

Quel niveau de langage ?

Commencer par du langage machine puis de l'assembleur est évidemment le plus formateur pour toucher du doigt le fonctionnement de la machine. De plus, c'est très motivant pour les élèves de rentrer en intimité avec cette machine qui semblait si obscure.

Malgré tout, il est bon de pouvoir utiliser notre langage dans toutes les disciplines qui nécessitent une mise en œuvre informatique. Du coup, il peut sembler mal aisé de développer des algorithmes de calculs, ou des simulations de Physique, en assembleur. Un langage de haut niveau restera plus polyvalent même si on sacrifie par ce choix une possibilité pour l'élève d'accéder au fonctionnement intime de la machine.

Typage et sucreries syntaxiques

Souvenons nous des vieux débats sur les langages qui faisaient rage dans les années 90, le typage était la grosse source de polémique. Les adeptes des typages forts faisaient des bonds quand on leur disait qu'il existait des langages dans lesquels on pouvait utiliser des variables sans même les déclarer... Par ailleurs, ça ne les choquait pas d'écrire des "for i in range(5):". Car soyons clairs, tout langage de haut niveau proposera des facilités d'écriture pour vous faire gagner du temps, que ce soit sur la déclaration des variables ou autre chose, masquant le fonctionnement rigoureux et détaillé de la machine.

Heureusement, ce débat est passé de mode et il s'avère finalement plus pédagogique qu'un élève comprenne que son programme a "planté" parce qu'il a voulu additionner un flottant avec une chaîne de caractères, plutôt que de le faire programmer dans un langage qui empêche totalement de le faire. Cela permet de poser comme clef de l'apprentissage une chose essentielle : la modélisation des données.

En somme, le typage n'est pas un critère de choix. Ce qu'on gagne d'un côté (rigueur, robustesse), on le perd de l'autre (compréhension plus fine de la modélisation).

En ce qui concerne la syntaxe et les sucreries syntaxiques que permettent les différents langages, il faut évacuer la question sans s'y arrêter. Les élèves ne sont pas idiots et ne sont jamais arrêtés par des problèmes de syntaxe : qu'on mette un ";" en fin de ligne ou des indentations en début de ligne, qu'importe. En fin de compte, tous les langages s'écrivent peu ou prou avec la même logique.

Sur ce point, Python est plutôt un mauvais choix car c'est finalement une rareté de donner un rôle syntaxique à l'indentation. De plus, certaines facilités d'écriture (range notamment) masquent aux élèves le fonctionnement de la machine alors que c'est précisément ce qu'ils ont besoin de comprendre.

Motivation et portabilité

Au final, les critères de choix ne s'avèrent pas vraiment décisifs et vous regrettez donc d'avoir lu cet article ;-). Mais restez encore un peu...

Et si une des bonnes clefs de lecture de ce problème était la motivation des élèves ? En effet, si nous voulons déclencher des appétences pour ce langage à ce premier contact, nous devons clairement envisager le problème sous cet angle, et c'est même peut-être finalement le seul argument qui vaille.

Or, à l'heure actuelle, pour motiver les élèves à la programmation, il faut évidemment pouvoir investir leurs terrains de jeux : Le smartphone, la tablette et le Web. Il est évident qu'un élève apprenant un langage "du lycée",

qui marche uniquement dans la salle informatique "du lycée", aura vite le sentiment qu'il n'en tirera aucun intérêt !

A l'inverse, si on lui propose un langage qui fonctionne spontanément (sans aucune installation) sur son smartphone ou le Web et sans aucune modification, il verra vite qu'il peut en faire son terrain de jeu et se l'approprier en créant ses applications ou ses jeux, qu'il pourra partager en un clic avec ses amis pour obtenir un maximum de likes !

A ce stade, si nous admettons que la motivation des élèves est primordiale, Python n'est pas du tout le meilleur choix pour enseigner la programmation au lycée à nos adolescents.

Une solution qui répond parfaitement à cette problématique de motivation est JavaScript ou ses dérivés. Pour ma part, je l'utilise depuis cinq ans avec mes élèves qui manifestent un réel enthousiasme quand ils apprennent que tout ce qu'ils vont faire va fonctionner partout, y compris sur le Web.

De plus, JavaScript a exactement la même logique que tous les langages multi-paradigmes très connus (si vous savez programmer en Python, vous saurez le faire en JavaScript). Et comme nous venons de l'analyser, ce type de langage est très adapté à un apprentissage au lycée.

Utilisation en classe

Un autre critère très important est celui de la facilité d'utilisation en classe comme à la maison.

Là encore, JavaScript est idéal car vous n'avez besoin d'aucune installation ! Ce qui veut dire que le jour où on vous change de salle informatique ou que votre Anaconda est malade d'une mise à jour mal digérée, qu'importe ! Avec JavaScript vous n'avez besoin de rien d'autre qu'un éditeur de texte (Notepad) et un navigateur internet (Firefox), ce qui est, vous en conviendrez, installé sur toutes les machines de la terre et même du lycée !

De même, si l'élève souhaite coder chez lui pour avancer ou tester de nouvelles choses, il ne sera pas freiné par l'étape d'installation du langage et de ses outils. Il pourra même programmer en vacances chez sa grand-mère sans difficulté et sans avoir à installer quoi que ce soit !

Quelles utilisations possibles ?

Pour ceux qui ne connaissent pas, résumons. Vous pouvez faire en JS (JavaScript) tout ce que vous faites en Python et très facilement : calculs mathématiques, simulations graphiques, jeux, etc. Le langage est très simple et l'utilisation des primitives graphiques est élémentaire. De plus, vous avez, comme en Python, des tonnes de "FrameWorks" pour faciliter la programmation dans différents domaines (3D, tensor flow, deep learning, etc.)

Conclusion

En résumé, Python est un non-choix d'arrière-garde car nous avons avant tout besoin d'accrocher les élèves avec un langage qu'ils ressentent comme un terrain de jeu possible, en exploitant leur appétit pour les technologies Web !

Pour finir, JavaScript et Python sont à peu près identiques quant à leurs caractéristiques principales, mais l'un parle aux élèves et l'autre aux professeurs de maths... Essayons donc de parler tous le même langage (bien choisi) !

Le débat est lancé !